(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 September 2001 (13.09.2001)

PCT

(10) International Publication Number
WO 01/67241 A1

(51) International Patent Classification[7]: G06F 9/45, H02H 3/05

(21) International Application Number: PCT/US01/06882

(22) International Filing Date: 5 March 2001 (05.03.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/519,234 6 March 2000 (06.03.2000) US

(71) Applicant: CONITA TECHNOLOGIES, INC. [US/US]; 1200 Main Street, Columbia, SC 29201 (US).

(72) Inventors: ULMER, Richard, M.; 1600 Shady Lane, Columbia, SC 29206 (US). PEEBLES, Edward; 318 Guild Hall Drive, Columbia, SC 29212 (US). SANDERS, Derek; 1412 Charbonneau Drive, Columbia, SC 29210 (US).

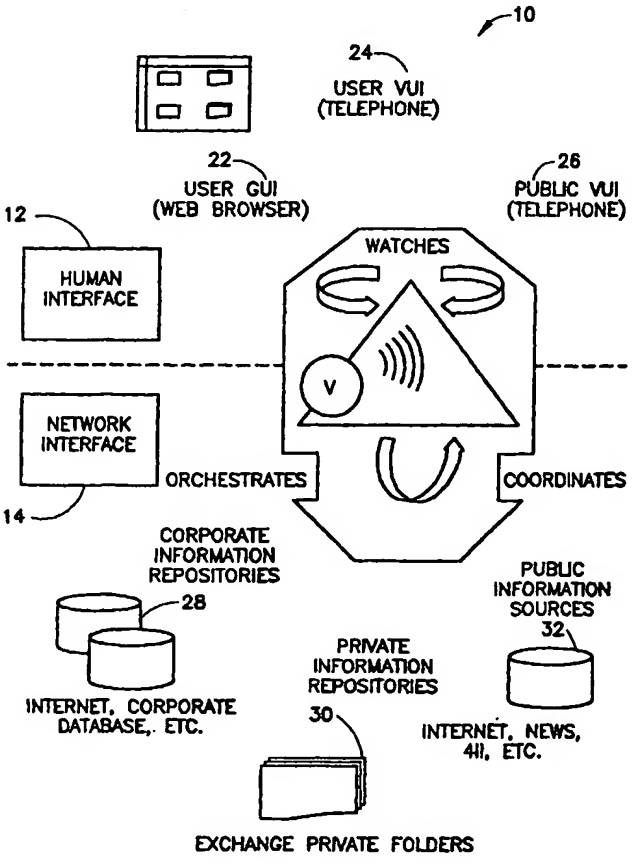
(74) Agent: MCFADDEN, Jeffrey, R.; Rhodes & Mason, P.L.L.C., P.O. Box 2974, Greensboro, NC 27402 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: VIRTUAL ASSISTANT ENGINE



(57) Abstract: A virtual assistant engine (10) for running a virtual assistant application (28, 30, 32), comprised of an interpreter for parsing, storing in a computer memory, and executing source code for a virtual assistant application, a scripting object that provides methods and properties for creating a virtual assistant application and an abstraction layer for interfacing with a speech recognition server (24), telephony hardware and a text to speech server, wherein the scripting object provides the interface between the abstraction layer and the virtual assistant application.

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

VIRTUAL ASSISTANT ENGINE

This application is related to application Serial No. _____, entitled Personal Virtual Assistant, Serial No. _____, entitled Virtual Assistant with Semantic Tagging, and Serial No. _____, entitled Virtual Assistant with Temporal Selectivity, which are filed simultaneously herewith, assigned to a common assignee, and are hereby incorporated by reference.

FIELD OF THE INVENTION

The present invention relates to a computer-based, virtual assistant engine.

BACKGROUND OF THE INVENTION

Mobile professionals, such as physicians, attorneys, sales representatives and other highly mobile professionals often find it difficult to communicate with clients, customers, colleagues and assistants. These mobile professionals travel frequently and are not accessible via a desk phone or traditional, wired computer network. They typically employ human assistants to relay important information, maintain their schedules and filter out all unnecessary interruptions. A virtual assistant is a computer application that allows the mobile professional to access personal, company, and public information, including contacts, schedules, and databases from any interactive device, such as telephone.

Previously, virtual assistant applications were hardcoded. In other words, a monolithic program written in the C++ programming language, for example, would implement all of the functions of and interfaces with the virtual assistant. Examples of such a virtual assistant application is described in U.S. Patent No. 5,652,789 to Miner, et al., and assigned to Wildfire Communications, Inc.

The problem with such prior art virtual assistant applications is that modifying or customizing the application is a difficult and time-consuming process. In order to make changes, a highly skilled computer programmer would be required to edit the source code, debug, recompile and link the edited source code. Then, the modified virtual assistant application had to be tested to ensure that it functioned as intended.

The present invention solves this problem by defining a virtual assistant application in terms of discourses, grammars, event handlers, and other components that instruct a virtual assistant engine as to how to execute of the application. This advantageously permits integration of the virtual assistant with other commercially available applications, including messaging applications and database management applications. The VA Application can be easily modified to satisfy specific requirements of a user.

SUMMARY OF INVENTION

The present invention relates to a virtual assistant system with many discrete features, each of which comprises a separate but related invention. Thus, one aspect of the present invention is a virtual assistant engine for running a virtual assistant application, comprised of an interpreter for parsing, storing in a computer memory, and executing virtual assistant definition language source code for a virtual assistant application, a scripting object that provides methods and properties for creating a virtual assistant application and an abstraction layer for interfacing with a speech recognition server, telephony hardware and a text to speech server, wherein the scripting object provides the interface between the abstraction layer and the virtual assistant application.

The interpreter is comprised of a parser for parsing the virtual assistant

definition language source code and storing the parsed virtual assistant definition language source code in the computer memory. The parser is constructed using the Purdue Compiler Constructor Tool Set.

The interpreter is further comprised of a state machine for executing the stored virtual assistant definition language source code. The state machine determines the tasks to be performed by the virtual assistant application responsive to input from the user. The state machine also manages a barge-in commands received from the user. In addition, the state machine manages external events responsive to output from the virtual assistant application, the output indicating that an external event has occurred, and is configured to cause the user to be notified of the occurrence of the external event. Examples of external events of which the virtual assistant user is notified are receipt of a telephone call, placing a telephone call, receipt of an electronic message, a meeting reminder, a task reminder, a change in a database and a change in monitored information.

The interpreter is further comprised of: a scripting host object, and a scripting engine, whereby the scripting host object interfaces with the scripting engine. The scripting engine executes scripts written in a scripting language, such as VBScript, JavaScript, Perl, REX and Python.

The interpreter is further comprised of a session object, which manages telephone calls to and from a virtual assistant application user. The session object is comprised of a call state manager for tracking the status, for example connected, on hold and in conference, of a telephone call to or from the virtual assistant application user. The session object is further comprised of a call object, for managing calls to the virtual assistant user from the virtual assistant and to the virtual assistant from the

virtual assistant user. The session object also is configured to generate and store in the computer memory a log of information about a virtual assistant application user session. The information log includes information about call statistics (for example, call duration, DNIS number, ANI number), call counters and call transcription (commands issued by the user and responses from virtual assistant.).

The interpreter is further comprised of a discourse manager, which activates the appropriate discourse responsive to input from the virtual assistant application user. The discourse manager also activates the appropriate grammar responsive to the active discourse.

The scripting object is configured to provide output to the user asynchronously. Such output is comprised of rendering text into speech or playing recorded prompts. The scripting object is further comprised of a management interface, which is configured to generate and store in the computer memory a log of information about virtual assistant application errors. The management interface also is configured to enable the management and configuration of a virtual assistant system by a system administrator.

The scripting object is further comprised of an interface for managing dynamic grammars. The management of dynamic grammars is comprised of, creating a user specific grammar when a virtual assistant application user session begins, storing the user specific grammar in the computer memory for use by a user during the user session, and deleting the user specific grammar from the computer memory when the user session ends. The user specific grammar is generated from a user-specified database.

The scripting object is further comprised of a state machine interface for

controlling a state machine for managing external events, and a call management interface for controlling a session object, which manages telephone calls to and from a virtual assistant application user.

The abstraction layer is comprised of a speech recognition module, a telephony module and a text to speech module. The speech recognition module is comprised of an interface between the scripting object and the speech recognition server. The telephony module is comprised of an interface between the scripting object and the telephony hardware, such as, an adapter for allowing electronic communication between the virtual assistant application and the virtual assistant user and a conference call adapter. The text to speech module is comprised of an interface between the scripting object and the text to speech server.

Other features and advantages will become apparent based on the following detailed description of the preferred embodiments and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an overview of the virtual assistant (VA) of the present invention;

FIG. 2 is a diagram of the VA Server;

FIG. 3 is a diagram of the VA Studio;

FIG. 4 is a diagram of the VA Engine conceptual model;

FIG. 5 is a diagram of the VA Manager conceptual model;

FIG. 6 is a screen shot of the Microsoft Management Console for managing the VA Server Manger;

FIG. 7 is a screen shot of a web page that uses Active Server Pages to manage the VA Server Manager;

FIG. 8 is a diagram of the component relationships of a VA Server Set;

FIG. 9 is a diagram of a relatively small VA system;

FIG. 10 is a diagram of a large VA system;

FIG. 11 is a diagram of a very large VA system;

FIG. 12 is a diagram of a VA discourse;

FIG. 13 is a diagram of the VA Discourse/Grammar Model;

FIG. 14 is a screen shot of the Main Application Window;

FIG. 15 is a screen shot of the Topic View Window;

FIG. 15 is a screen shot of the Task View Window;

FIG. 16 is a screen shot of the Prompt Properties Dialogue Box;

FIG. 17 is a screen shot of the an Expanded Tree View of the Menu_Error Prompt Group; and

FIG. 18 is a screen shot of the Prompt Group Properties Dialogue Box.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The subheadings used herein are meant only so as to aid the reader and are not meant to be limiting or controlling upon the invention. Generally, the contents of each subheading are readily utilized in the other subheadings.

Overview

Mobile professionals, such as physicians, attorneys, sales representatives and

other highly mobile professionals often find it difficult to communicate with clients, customers, colleagues and assistants. These mobile professionals travel frequently and are not accessible via a desk phone or traditional, wired computer network. They typically employ assistants to relay important information, maintain their schedules and filter out all unnecessary interruptions. The virtual assistant of the present invention allows the mobile professional to access personal, company, and public information, including contacts, schedules, and databases from any interactive device, such as telephone.

The virtual assistant ("VA") system of the present invention is comprised of two main components: (1) the VA Server, which is built on a Windows NT telephony server platform, and (2) the VA Studio, which allows skilled information technology professionals to develop VA applications that interface with electronic messaging systems, such as Microsoft Exchange and Lotus Notes. The VA Server is a component of the Service Deployment Environment ("SDE"), which is discussed in more detail below. The VA Studio is a component of the Service Creation Environment ("SCE"), which is also discussed in more detail below.

As shown in Figure 1, the VA Server 10 is comprised of a human interface 12 and a network interface 14 for handling calls and providing automated access to information to corporate 28, private 30 and pubic 32 information repositories and sources. The human interface 12 is comprised of a graphical user interface 22, which may be a web browser, a subscriber (or user) voice user interface 24, generally accessed by a telephone, and a public voice user interface 26. The virtual assistant allows a user to use a voice interactive device, such as a telephone, either wired or wireless, to access and update such information. The VA Server also manages all incoming communications by sorting, prioritizing, and filtering such communications, while providing notice to the user of important messages and events.

VA Server

As seen in Figure 2, a core component of the VA Server 40 is the voice-enabled Virtual Machine 42, which is also referred to as the VA Engine. The VA Engine receives spoken commands, interprets and executes them. The VA Engine supports a COM interface 44, which in turn enables VA applications to provide voice

access to network applications.

The VA Engine also supports a telephony interface 46 to voice messaging 52 and private branch exchange systems 54, enabling third-party systems to be integrated with the VA Server.

The VA Server conforms to Windows NT telephony and speech interface specifications. The voice-messaging interface 56 supports the VPIM (Voice Profile for Internet Mail) standard, and provides a gateway between proprietary voice messaging systems and VA Server.

The VA system management services provide operations, administration and maintenance capability (OA&M) 60. The OA&M applications also provide a Simple Network Management Protocol ("SNMP") interface to third party management applications, for example, HP Openview and CA Unicenter.

In the preferred embodiment, the VA Server is operable on Windows NT Server, release 4.0 or higher, in both single and multiprocessor configurations. Those skilled in the art, however, recognize that the VA Server can be ported to other computing platforms. Multiple systems may be clustered together to support higher system workloads and fail-safe operation.

VA Application Suite

The VA Application Suite, in the preferred embodiment, is compatible with a messaging server 62, such as Microsoft Exchange/Outlook. The VA's architecture, however, advantageously permits integration with other commercially available and customized messaging applications. The VA Application can be easily modified to satisfy specific requirements of a user. The basic functions of the VA Application include:

Messaging – voice-mail, e-mail, and faxes

Contact Management - scheduling, planning, group calendar, contact and referral organization

Call Control – remote users to perform conference calling and call management; notification and forwarding features allow remote users to be contacted immediately by phone/pager when they receive specific voice-mails, e-mails, faxes, or pages

Internet Applications – users can access and internet via an internet server 64 and obtain public information such as weather, travel, financial, competitive data and news

Intranet Applications – users can remotely access information contained on a corporate network (inside the company firewall) using the VA, for example, customer data, shipping and inventory information, sales reports, and financial data, or any information on a database server 66, including SQL databases such as Oracle or Informix.

Customer Relationship Management applications – the VA Server integrates with commercially available customer relationship management (CRM) software applications 70, such as Siebel, Pivotal, Sales Logix and Onyx.

VA Studio

As seen in Figure 3, the VA Studio 80 is comprised of a grammar generator 82 and a publishing toolkit 84. The VA Studio allows a user to create, modify and debug applications that run on the VA Server 40 without requiring the user to be skilled in the complexities of the underlying components of the VA Server, such as the speech recognition engine, text to speech engine, switch control and unified messaging.

VA Studio employs a graphical user interface (GUI) application that runs on a Windows NT workstation. It allows developers to create projects, each of which defines a VA application. VA Studio is a multiple document interface (MDI) application that follows the workspace-based model.

The VA Studio follows the Microsoft Component Object Model (COM). VA applications are developed using Active Scripting languages such as VBScript and JavaScript, thus enabling integration with a variety of third party components. The VA applications created with the VA studio will include voice query to SQL databases, message stores, business logic and mainframe applications.

VA applications are composed of discourses and resources. Discourses are the context of conversations between a user and the VA. Resources are items like voice prompts and dictionaries. A developer can utilize the VA Studio Wizard to generate a "skeleton" VA application template. Application templates consist of packages of predefined discourses and resources. Discourses are the context of conversations

between a user and the VA. Resources are items like voice prompts and dictionaries. Once a VA application template is generated, the application is further customized using any supported Active Scripting languages.

After writing the VA application, it is then submitted to the build process. During the build process, VA Studio checks for dialog errors, builds a master intermediate grammar and builds a master lexicon. Once compiled and error-free the application is ready to be published.

When an application is published, it is transported from the VA Studio to the VA Server. The VA Server allows a scripted application to access services such as voice mail, databases, and telephony equipment.

A VA application is created, modified, debugged and tested using the VA Studio. The completed application is then automatically installed and configured to run on the VA Server, which enables the VA application to take incoming calls and provide access to both public and private information.

Platform Overview

An Introduction to Virtual Assistant Applications

A VA application allows a user to manage electronic communications and access his or her business's computer resources through a telephone. Using speech recognition and text-to-speech technology, the VA communicates with callers in spoken English. By calling into the VA on a standard telephone, a user can perform functions such as the following:

- Sending and receiving voice mail messages
- Checking, replying to, and forwarding email messages
- Looking up phone numbers and addresses in an electronic address book
- Accessing information in a company database
- Accessing information on the World Wide Web

In addition, the VA can perform many of the functions of a personal secretary, such as the following:

- Informing the user via pager when new voice and email messages arrive
- Filtering incoming voice mail, email, and pages as instructed by the user

- Automatically dialing phone numbers

In the preferred embodiment, the VA performs the above functions by interfacing with a company's Microsoft Exchange server. This application, in effect, allows users to use their desktop Outlook software over the telephone.

The VA software includes a development platform (the SCE) and run-time platform (the SDE), which can host a variety of different VA's. The SDE provides the core components necessary for the functionality of a VA: a telephony interface, speech recognition facilities, a text-to-speech engine, interfaces with databases and mail servers, and an administrative framework in which the assistant applications will run. The SCE also includes development tools that programmers can use to create custom VA applications.

VA Platform Components

As discussed above, the VA Platform consists of three main components:

- The Service Deployment Environment (SDE)
- Virtual Assistant Applications
- The Service Creation Environment (SCE)

The function of each of these components can be understood using a World Wide Web analogy. The SDE functions like a web server, providing connections with the network and telephone system, controlling the execution of VA applications, and providing resources such as text-to-speech and voice recognition engines that will be accessed by the applications that run on it.

The VA applications are analogous to web pages, determining the content that will be presented and controlling the interactions with the user. A VA application uses scripting languages such as VBScript, JavaScript, and Perl, so that developers can add significant functionality to a VA, such as performing mathematical calculations, processing text, and calling ActiveX and COM objects.

Just as Microsoft Front Page and Netscape Composer are used to create web pages, the SCE is the development environment used to create the VA applications. The main component of the SCE is the VA Studio application, which is based on the Microsoft Visual Studio paradigm and provides a graphical environment with a

variety of tools that can be used to create, debug, and publish applications that are run on the SDE. The SCE also includes a set of COM objects that can be used in applications to perform functions such as checking email, reading from a database, and manipulating sound files.

The SDE Service Processes

The Service Deployment Environment consists of eight processes that run simultaneously and perform the functions necessary to support a VA application. In the preferred embodiment, each of these SDE components runs as a Windows NT Service or background process.

Although they may all run on the same hardware platform, for large VA implementations the components can be distributed across several servers and communicate over the network. Such distribution can allow, for example, one server to be dedicated to performing voice recognition functions while another supports the VA Engine that actually runs the applications. When multiple VA components are distributed across multiple machines, these machines are collectively termed a *VA server set.*

The VA Engine

As illustrated in Figure 4, the VA Engine 100 is the virtual machine on which a VA application 102 runs. Based on the application's instructions, the VA Engine uses its telephony interface 104 to communicate with the user 106 and its speech interface 110 to recognize speech into text and translate text into speech. The VA Engine connects to an Active Scripting Engine 112 to execute the scripts contained in the VA application, and it also communicates with administrative processes such as the VA Server 114 and VA Manager 116.

A VA Engine process can support user interaction over only one telephone line, but multiple VA Engines can be run simultaneously on a single platform. If the VA platform is connected to more than one telephone line, then a separate VA Engine will be running for each incoming line.

The Text-to-Speech (TTS) Server

The Text-to-Speech Server receives text from other components, translates it into speech (that is, into a sound file), and returns it to the requesting component. This speech translation service is isolated in a separate component to improve performance and to allow for TTS vendor-independence. The preferred embodiment uses the AcuVoice TTS system, but the platform can be easily modified to support a TTS engine from a different vendor. Only the TTS Server component would have to be modified for such a customization, not the entire platform.

Multiple VA Engines can use the same TTS Server process, and more than one TTS Server can be running at the same site, allowing translation services to be distributed across multiple machines for load-balancing.

The Recognition Server

The Recognition Server 122 receives sound files from other components, attempts to recognize them as speech, and returns the recognized text. Like the TTS server, the Recognition Server is a component that isolates speech-recognition functions from the rest of the VA platform. The server provides an interface to a third-party voice recognition engine (in the preferred embodiment, Nuance) that can be changed to a different vendor's brand without requiring the entire VA platform to be modified.

Multiple VA Engines can use the same Recognition Server process, and more than one Recognition Server can be running simultaneously.

Recognition Server Sub-Processes

The Recognition Server process requires three additional processes to be running:

The Resource Manager: The Resource Manager is a management process that automatically load-balances requests when more than one instance of the Recognition Server is running. Rather than making recognition requests to a particular Recognition Server, the VA Engine makes the request to the Resource Manager, which forwards it to the first available Recognition Server.

The Compilation Server: The Compilation Server compiles dynamic grammars.

The License Manager: The License Manager server runs continually in the background and dispenses licenses to all requesting components. Only one license manager need run in a single server set, but no Recognition Server components can launch unless the license manager is already running.

In the preferred embodiment, all of the sub-processes of the Recognition Server are recommended only for Nuance brand Recognition Servers. If a user uses different speech recognition software, a different set of processes may be needed.

The VA Server

- The VA Server 114 performs persistent VA functions that occur even when no user is connected to a VA application. These functions include the following:
- Monitoring external sources such as email boxes, databases, and web sites for events (e.g. a new mail message arrives or a database field is updated)
- Applying rules and filters to external source events to determine whether the VA system should take any actions
- Paging users when specified events occur

Only one VA Server can run on a system, but a single VA Server can provide persistent services to multiple VA Engines running both locally and on remote systems.

The VA Manager

As illustrated in Figure 5, each system that is running one or more VA components should also be running the VA Manager application 116. This application creates and monitors all VA components that are active on the system, and it provides management interfaces that are used for the following purposes:

- Configuration (both at start-up and during run-time)
- Signaling of events such as errors and informational messages
- Logging of events
- Logging of each call received by the VA applications running on the system
- Performance monitoring (through an interface to Window NT's **Perfmon**

utility)

The VA Manager provides the interface through which the VA Server Manager 130 communicates with all systems in use at the site.

The VA Server Manager

The VA Server Manager 130 provides a single point of control for all of the processes and servers being used in a VA server set. It communicates with the VA Manager 116 running on each VA server in the set and, through this interface, allows an administrator to use a single system to manage the entire site.

There are two ways an administrator can connect with the VA Server Manager application:

- **Using the Microsoft Management Console (MMC):** As illustrated in Figure 6, the VA software includes an MMC snap-in component 140 that allows the VA Server Manager services (and, thereby, the entire VA site) to be managed from the Microsoft Management Console application.
- **Using an Administrative Web Page:** The VA software also includes an administrative web page 142 that uses Active Server Pages to interface with the VA Server Manager service, allowing an administrator to manage the site through a standard web browser.

Returning to Figure 5, the VA Server Manager 130 monitors all of the VA components (such as Recognition Servers 132, TTS Servers 134, and VA Engines 136) running on all the systems within the server set, and it can be configured to page the system administrator with an alert if components fail or other system-critical events occur.

Additional VA Platform Components

In addition to the service processes, the following components are used on the VA platform.

The VA Database

The VA Server Manager process uses a Microsoft MSDE database to store

configuration parameters and platform logs. The MSDE database engine is installed automatically as part of the VA platform install, and the required tables and initial data are created during the installation routine.

The VA Server Manager uses a COM object called **DBManager** to communicate with the database. This object is created automatically at start-up by the VA Server Manager and provides a set of application programming interfaces (API's) that other VA components can use to retrieve configuration information and log data. In addition, the DBManager object automatically handles version checking, database restoration, and other database management functions.

The VA Web Server

In the preferred embodiment, as illustrated in Figure 7, the VA platform uses a Microsoft IIS Web Server to support browser-based administrative utilities. For example, the **VA Logging Tool** is used by the administrator to view and manage system logs.

VA Shared Directories

The VA software uses a set of shared directories for storing files necessary for platform operations. In a multi-server implementation, these shares are stored on a central server (the same server that hosts the VA Server Manager process) and can be accessed by all the systems in the server set. The shared directories used by the VA platform are described in the table below.

Table 1-1: VA Platform Shared Directories

Directory	Description
%*conitava*%\VAApplications*	Used to store the source files for the applications that will run on the platform
%*conitava*%\VALogs	Used to store application logs
%*conitava*%\VAUsers	Used to store information about VA users
%*conitava*%\VAUtterances	Used to store temporary sound files containing the commands spoken by VA users

* %*conitava*% represents the base path under which the VA platform software was installed. By default, this path is *c:\Program Files\Conita Virtual Assistant.*

VA Platform Configurations

The service processes that make up the VA platform either can be run on a single server (a VA platform server) or can be distributed across multiple servers (a VA platform server set). A single-server implementation is adequate for small companies that need to support only a few incoming VA calls at a time. For larger companies, however, a server set implementation will be necessary for load balancing.

As illustrated in Figure 8, when the VA platform is distributed across multiple servers, one node in the server set is designated the *Server Set Controller Node* 150. As the platform's primary server, the Server Set Controller Node will host the following components:

- The VA Server Manager service 152
- The VA DBManager service 154 and the VA database 156
- The IIS web-server
- Shared directories that will be used by all the servers in the server set to store logs, utterance files, application files, and user information

Each secondary node 160 in the set will host one or more instances of VA Engines 162, TTS Servers 164, and/or Recognition Servers 166. These processes will be monitored by a VA Manager process 170 on each server, which will in turn communicate with the VA Server Manager 172 on the Server Set Controller Node 150. In single-server implementations, the lone server is configured as the controller node, hosting the database, web-server, and VA Server Manager process along with all other VA services.

Scaling a VA Implementation

The way a business configures its VA platform will depend on the number of users who will be interacting with the VA application. As illustrated in Figure 9, for smaller sites, all the VA components can be run on a single server 180. Such a site could support several incoming telephone lines 182, allowing up to multiple instances of the VA application to be running simultaneously.

For larger sites that need to support many simultaneous VA application

sessions, the VA components can be distributed across multiple systems. As illustrated in Figure 10, a medium-sized company may, for instance, use a six-server rack 184, with two of the servers running VA Engines 186a, 186b, two servers running Recognition Servers 190a, 190b, one running VA Servers 192, and one running TTS Servers 194.

A large organization may require even more scalability. As illustrated in Figure 11, to support a public switch 196 with 32 incoming T1 lines 200, the site may use upwards of eight systems for VA Engines 202, sixteen for Recognition Servers 204a, 204b, four for VA Servers 206, and four for TTS Servers 206.

Duties of the VA Administrator

The duties of the VA Virtual Assistant platform administrator include the following tasks:

- Preparing the server(s) for installation of the VA platform software
- Installing the VA software on the systems
- Ensuring that the application software can communicate with the telephone system and other hosts such as a Microsoft Exchange server
- Configuring Microsoft Exchange to support VA users
- Using the VA management interfaces to manage the systems in the server set, start and stop the VA services, and run VA applications
- Monitoring the platform interfaces and error logs
- Maintaining the VA database
- Managing VA user accounts

In order to perform the above duties, a VA administrator needs to have experience with the following software packages:

- Windows NT
- Microsoft Exchange Server
- Microsoft Internet Information Server (IIS)
- Microsoft MSDE or SQL Server databases

Fundamentals of a VA Application

What is claimed is:

1. A virtual assistant engine for running a virtual assistant application, comprised of:

an interpreter for parsing, storing in a computer memory, and executing source code for a virtual assistant application;

a scripting object that provides methods and properties for creating a virtual assistant application; and

an abstraction layer for interfacing with a speech recognition server, telephony hardware and a text to speech server, wherein the scripting object provides the interface between the abstraction layer and the virtual assistant application.

2. The virtual assistant engine of claim 1, wherein the interpreter is comprised of a parser for parsing the source code and storing the parsed source code in the computer memory.

3. The virtual assistant engine of claim 2, wherein the parser is constructed using the Purdue Compiler Constructor Tool Set.

4. The virtual assistant engine of claim 2, wherein the interpreter is further comprised of a state machine for executing the stored source code.

5. The virtual assistant engine of claim 4, wherein the state machine determines the tasks to be performed by the virtual assistant application responsive to input from the user.

6. The virtual assistant engine of claim 4, wherein the state machine manages a barge-in command received from the user.

7. The virtual assistant engine of claim 4, wherein the state machine manages external events responsive to output from the virtual assistant application, the output indicating that an external event has occurred, and is configured to cause the user to be notified of the occurrence of the external event.

8. The virtual assistant engine of claim 7, wherein the external event is selected from the group consisting of receipt of a telephone call, placement a telephone call, receipt of an electronic message, a meeting reminder, a task reminder, a change in a database and a change in monitored information.

9. The virtual assistant engine of claim 4, wherein the interpreter is further comprised of:

a scripting host object; and

a scripting engine, whereby the scripting host object interfaces with the scripting engine.

10. The virtual assistant engine of claim 9, wherein the scripting engine executes scripts written in a scripting language selected from the group consisting of VBScript, JavaScript, Perl, REX and Python.

11. The virtual assistant engine of claim 9, wherein the interpreter is

further comprised of a session object.

12. The virtual assistant engine of claim 11, wherein the session object manages telephone calls to and from a virtual assistant application user.

13. The virtual assistant engine of claim 11, wherein the session object is comprised of a call state manager for tracking the status of a telephone call to or from the virtual assistant application user.

14. The virtual assistant engine of claim 13, wherein the status of the telephone call is selected from the group consisting of connected, on hold and in conference.

15. The virtual assistant engine of claim 11, wherein the session object is further comprised of a call object for managing calls to the virtual assistant user from the virtual assistant and to the virtual assistant from the virtual assistant user.

16. The virtual assistant engine of claim 11, wherein the session object is configured to generate and store in the computer memory a log of information about a virtual assistant application user session.

17. The virtual assistant engine of claim 16, wherein the information log is comprised of call statistics information, the call statistic information being comprised of information about the length of the user session, the DNIS number and the ANI number.

18. The virtual assistant engine of claim 16, wherein the information log is comprised of call transcription information, the call transcription information being comprised of information about commands issued by the virtual assistant user and the responses from virtual assistant to the commands during the user session.

19. The virtual assistant engine of claim 16, wherein the information log is comprised of speech recognition information, the speech recognition information being comprised of latency information and speech recognition performance.

20. The virtual assistant engine of claim 19, wherein the latency information is comprised of information about the amount of time taken by the virtual assistant to respond to commands from the user during the user session.

21. The virtual assistant engine of claim 19, wherein the speech recognition performance information is comprised of information about speech recognition accuracy and the amount of processing time for speech recognition.

22. The virtual assistant engine of claim 16, wherein the information log is comprised of system information, the system information being comprised of the amount of time spent by the virtual assistant receiving input from the user, processing the input received from the user, and providing output to the user.

23. The virtual assistant engine of claim 11, wherein the interpreter is further comprised of a discourse manager.

24. The virtual assistant engine of claim 23, wherein the discourse manager activates the appropriate discourse responsive to input from the virtual assistant application user.

25. The virtual assistant engine of claim 24, wherein the discourse manager activates the appropriate grammar responsive to the active discourse.

26. The virtual assistant engine of claim 1, wherein the scripting object is configured to provide output to the user asynchronously.

27. The virtual assistant engine of claim 26, wherein the output is generated by rendering text into speech.

28. The virtual assistant engine of claim 26, wherein the output is generated by playing recorded prompts.

29. The virtual assistant engine of claim 26, wherein the scripting object is further comprised of a management interface.

30. The virtual assistant engine of claim 29, wherein the management interface is configured to generate and store in the computer memory a log of information about virtual assistant application errors that occurred during a virtual assistant user session.

31. The virtual assistant engine of claim 29, wherein the management interface is configured to enable the management and configuration of a virtual assistant system by a system administrator.

32. The virtual assistant engine of claim 29, wherein the scripting object is further comprised of an interface for managing dynamic grammars.

33. The virtual assistant engine of claim 32, wherein the management of dynamic grammars is comprised of:

creating a user specific grammar when a virtual assistant application user session begins;

storing the user specific grammar in the computer memory for use by a user during the user session; and

deleting the user specific grammar from the computer memory when the user session ends.

34. The virtual assistant engine of claim 33, wherein the user specific grammar is generated from a user specified database.

35. The virtual assistant engine of claim 32, wherein the scripting object is further comprised of a state machine interface for controlling a state machine for managing external events.

36. The virtual assistant engine of claim 29, wherein the scripting object is further comprised of call management interface for controlling a session object,

wherein the session object manages telephone calls to and from a virtual assistant application user.

37. The virtual assistant engine of claim 1, wherein the abstraction layer is comprised of a speech recognition module, a telephony module and a text to speech module.

38. The virtual assistant engine of claim 37, wherein the speech recognition module is comprised of an interface between the scripting object and the speech recognition server.

39. The virtual assistant engine of claim 37, wherein the telephony module is comprised of an interface between the scripting object and the telephony hardware.

40. The virtual assistant engine of claim 39, wherein the telephony hardware is comprised of an adapter for allowing electronic communication between the virtual assistant application and the virtual assistant user, and a conference call adapter.

41. The virtual assistant engine of claim 37, where in the text to speech module is comprised of an interface between the scripting object and the text to speech server.

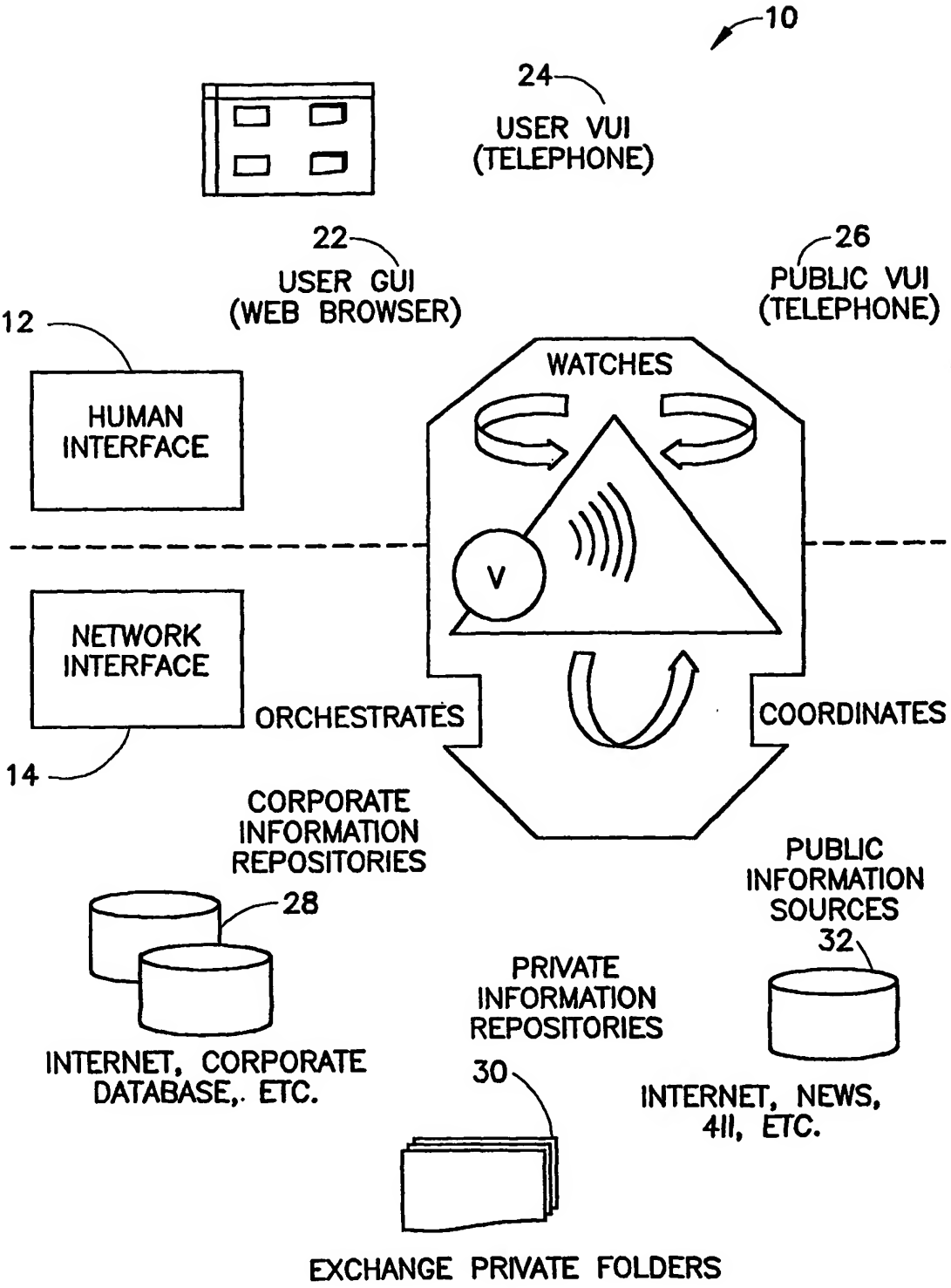

10
24
USER VUI
(TELEPHONE)
22
USER GUI
(WEB BROWSER)
26
PUBLIC VUI
(TELEPHONE)
12
HUMAN
INTERFACE
WATCHES
V
NETWORK
INTERFACE
14
ORCHESTRATES
COORDINATES
CORPORATE
INFORMATION
REPOSITORIES
28
INTERNET, CORPORATE
DATABASE,. ETC.
PUBLIC
INFORMATION
SOURCES
32
PRIVATE
INFORMATION
REPOSITORIES
30
INTERNET, NEWS,
411, ETC.
EXCHANGE PRIVATE FOLDERS

FIG. 1

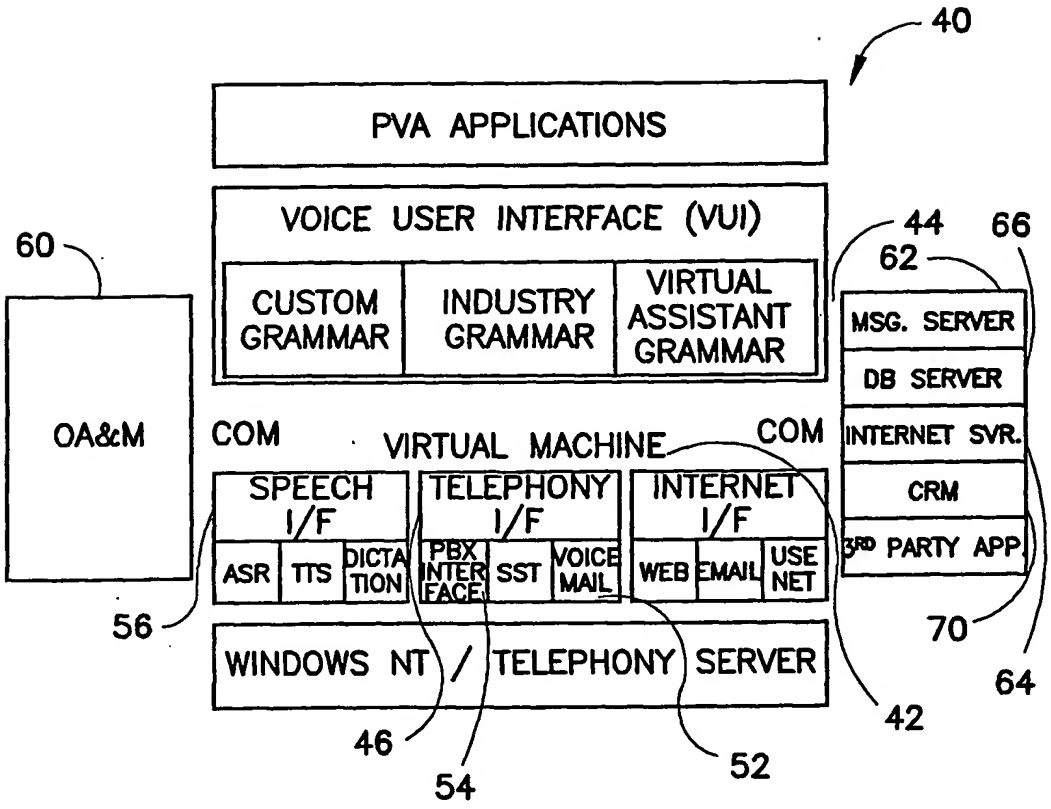

40
PVA APPLICATIONS
VOICE USER INTERFACE (VUI)
CUSTOM GRAMMAR
INDUSTRY GRAMMAR
VIRTUAL ASSISTANT GRAMMAR
60
OA&M
COM
VIRTUAL MACHINE
COM
SPEECH I/F
ASR
TTS
DICTATION
TELEPHONY I/F
PBX INTERFACE
SST
VOICE MAIL
INTERNET I/F
WEB
EMAIL
USENET
WINDOWS NT / TELEPHONY SERVER
44
66
62
MSG. SERVER
DB SERVER
INTERNET SVR.
CRM
3RD PARTY APP.
56
70
64
42
46
54
52

FIG. 2

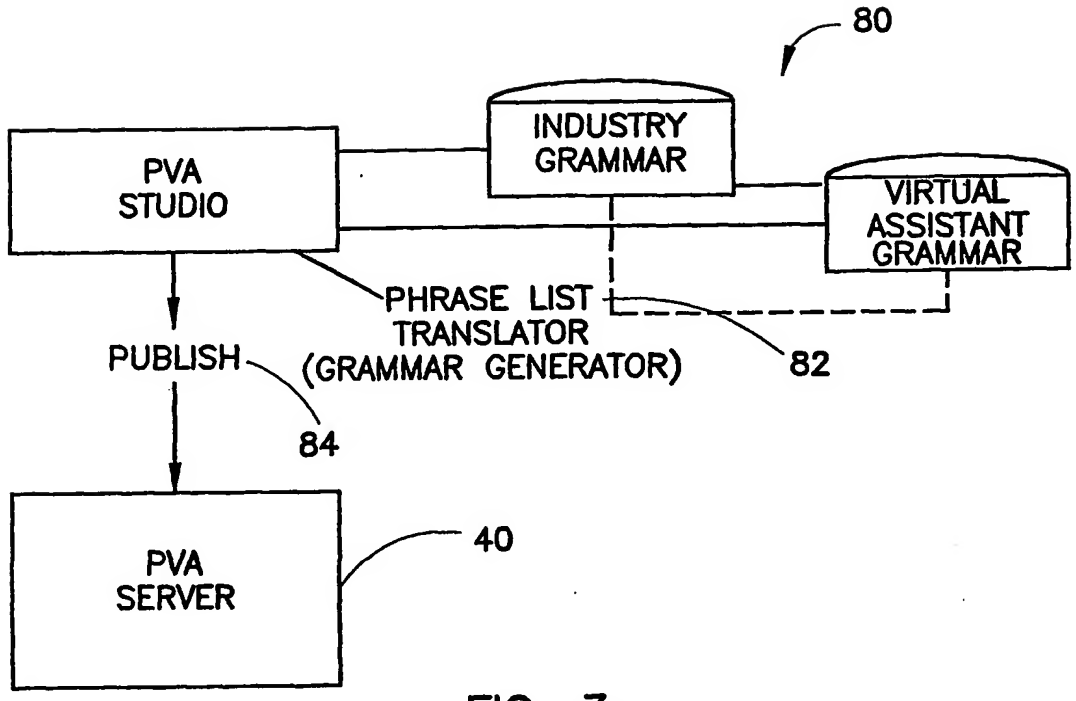

80
INDUSTRY GRAMMAR
PVA STUDIO
VIRTUAL ASSISTANT GRAMMAR
PHRASE LIST TRANSLATOR (GRAMMAR GENERATOR)
82
PUBLISH
84
PVA SERVER
40

FIG. 3

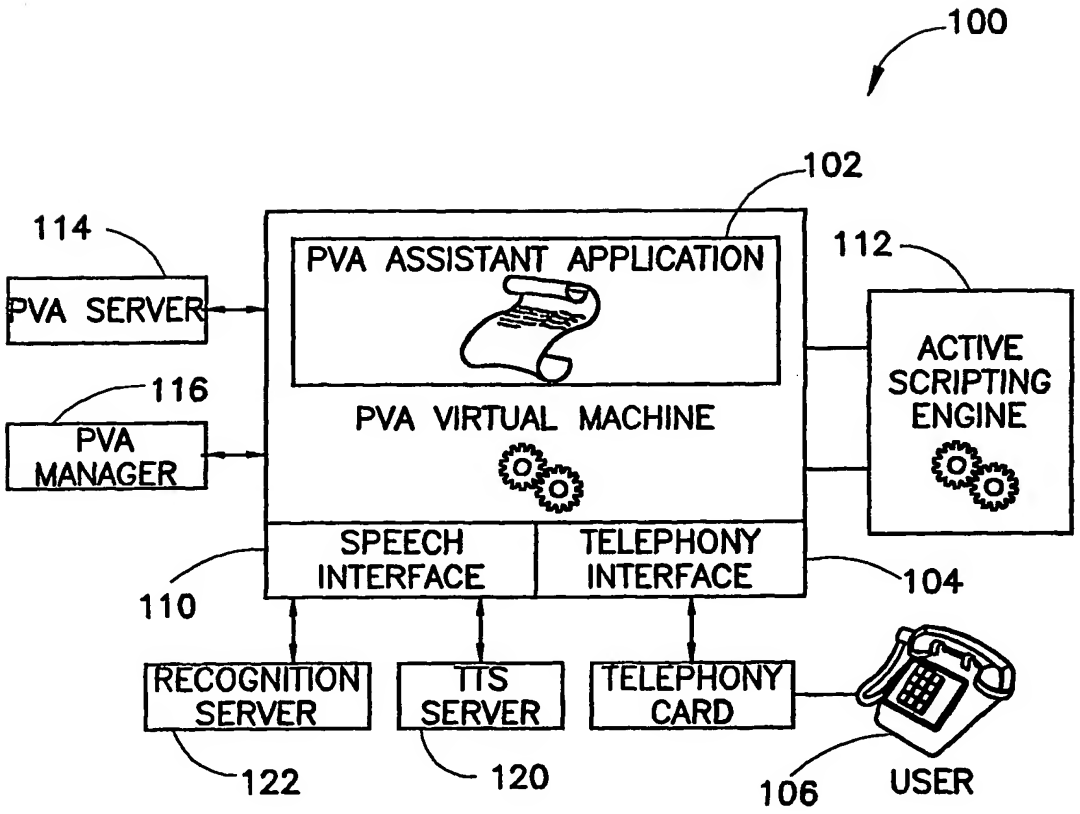

100
102
114
PVA SERVER
PVA ASSISTANT APPLICATION
112
ACTIVE SCRIPTING ENGINE
116
PVA MANAGER
PVA VIRTUAL MACHINE
SPEECH INTERFACE
TELEPHONY INTERFACE
110
104
RECOGNITION SERVER
TTS SERVER
TELEPHONY CARD
122
120
106
USER

FIG. 4

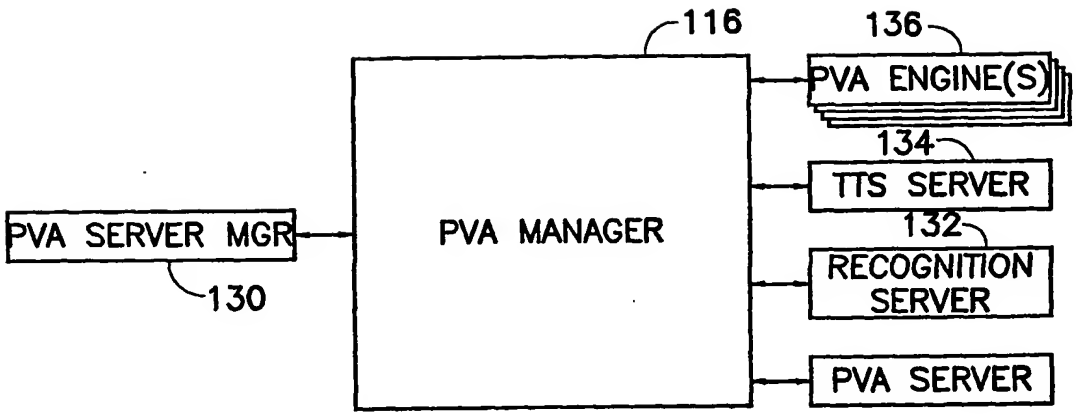

116
136
PVA ENGINE(S)
134
TTS SERVER
132
RECOGNITION SERVER
PVA SERVER
PVA SERVER MGR
130
PVA MANAGER

FIG. 5

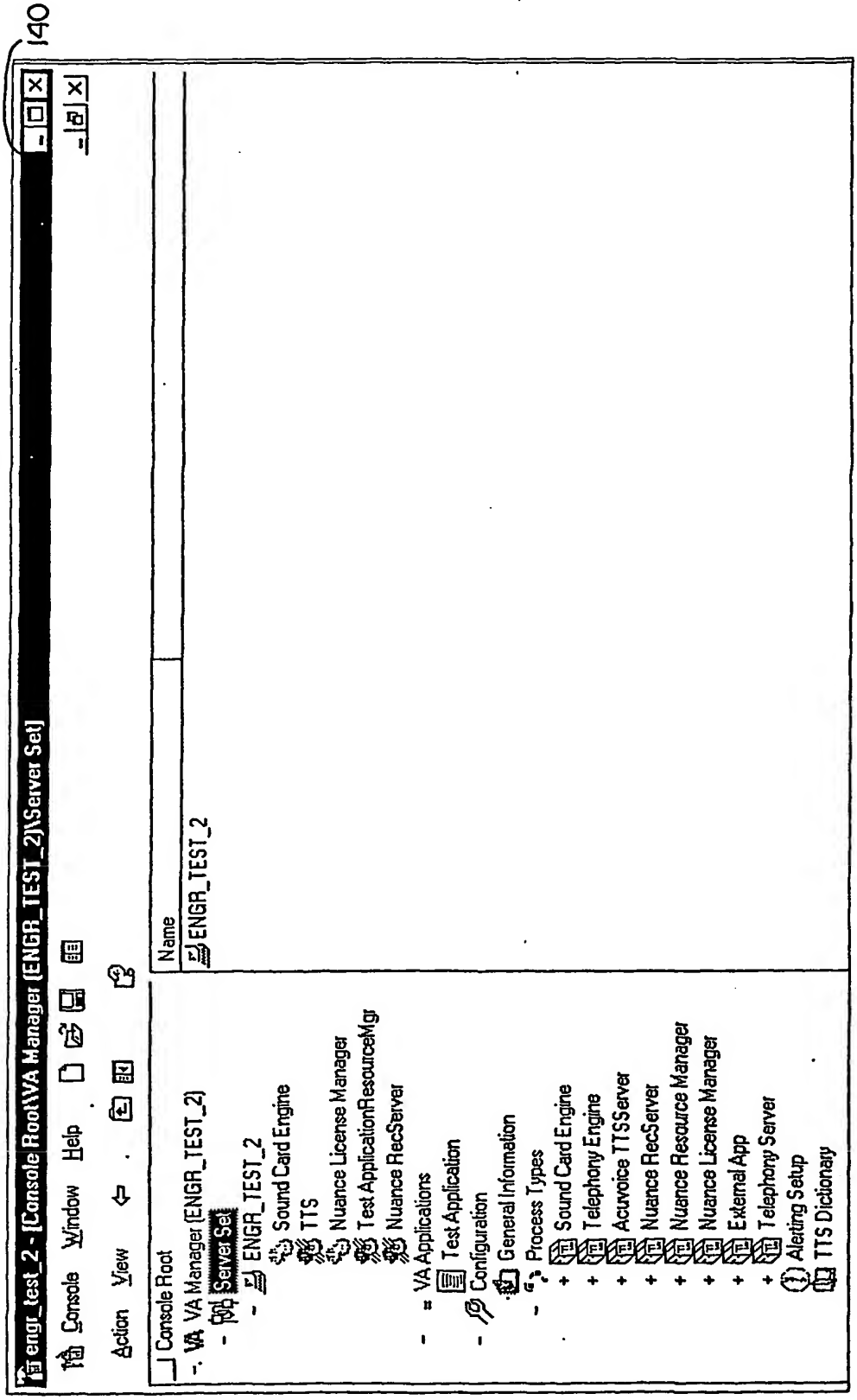
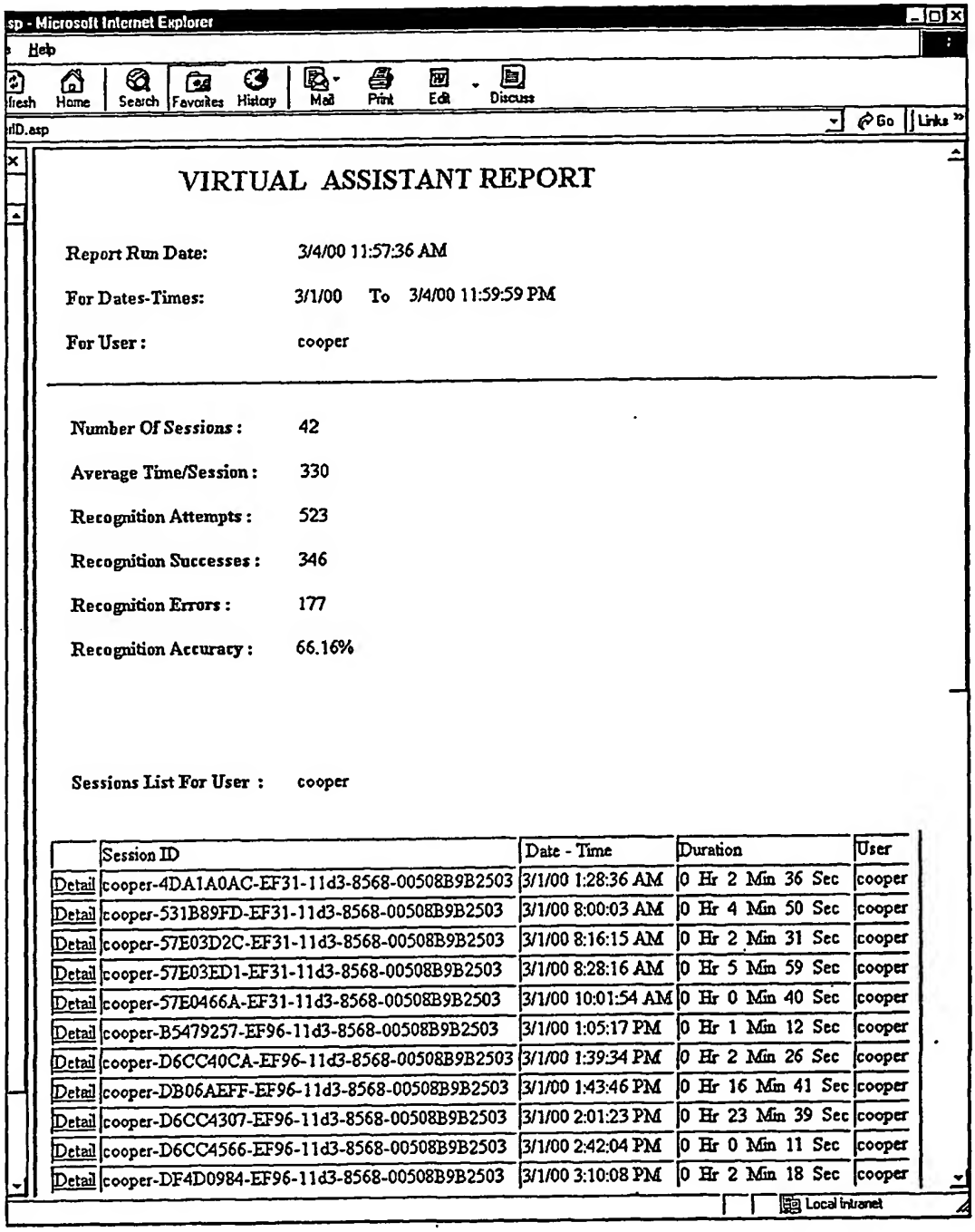

140
engr_test_2 - [Console Root\VA Manager (ENGR_TEST_2)\Server Set]
Console Window Help
Action View
Console Root
VA Manager (ENGR_TEST_2)
Server Set
ENGR_TEST_2
Sound Card Engine
TTS
Nuance License Manager
Test ApplicationResourceMgr
Nuance RecServer
VA Applications
Test Application
Configuration
General Information
Process Types
Sound Card Engine
Telephony Engine
Acuvoice TTSServer
Nuance RecServer
Nuance Resource Manager
Nuance License Manager
External App
Telephony Server
Alerting Setup
TTS Dictionary
Name
ENGR_TEST_2

FIG. 6

FIG. 7



	Session ID	Date - Time	Duration	User
Detail	cooper-4DA1A0AC-EF31-11d3-8568-00508B9B2503	3/1/00 1:28:36 AM	0 Hr 2 Min 36 Sec	cooper
Detail	cooper-531B89FD-EF31-11d3-8568-00508B9B2503	3/1/00 8:00:03 AM	0 Hr 4 Min 50 Sec	cooper
Detail	cooper-57E03D2C-EF31-11d3-8568-00508B9B2503	3/1/00 8:16:15 AM	0 Hr 2 Min 31 Sec	cooper
Detail	cooper-57E03ED1-EF31-11d3-8568-00508B9B2503	3/1/00 8:28:16 AM	0 Hr 5 Min 59 Sec	cooper
Detail	cooper-57E0466A-EF31-11d3-8568-00508B9B2503	3/1/00 10:01:54 AM	0 Hr 0 Min 40 Sec	cooper
Detail	cooper-B5479257-EF96-11d3-8568-00508B9B2503	3/1/00 1:05:17 PM	0 Hr 1 Min 12 Sec	cooper
Detail	cooper-D6CC40CA-EF96-11d3-8568-00508B9B2503	3/1/00 1:39:34 PM	0 Hr 2 Min 26 Sec	cooper
Detail	cooper-DB06AEFF-EF96-11d3-8568-00508B9B2503	3/1/00 1:43:46 PM	0 Hr 16 Min 41 Sec	cooper
Detail	cooper-D6CC4307-EF96-11d3-8568-00508B9B2503	3/1/00 2:01:23 PM	0 Hr 23 Min 39 Sec	cooper
Detail	cooper-D6CC4566-EF96-11d3-8568-00508B9B2503	3/1/00 2:42:04 PM	0 Hr 0 Min 11 Sec	cooper
Detail	cooper-DF4D0984-EF96-11d3-8568-00508B9B2503	3/1/00 3:10:08 PM	0 Hr 2 Min 18 Sec	cooper

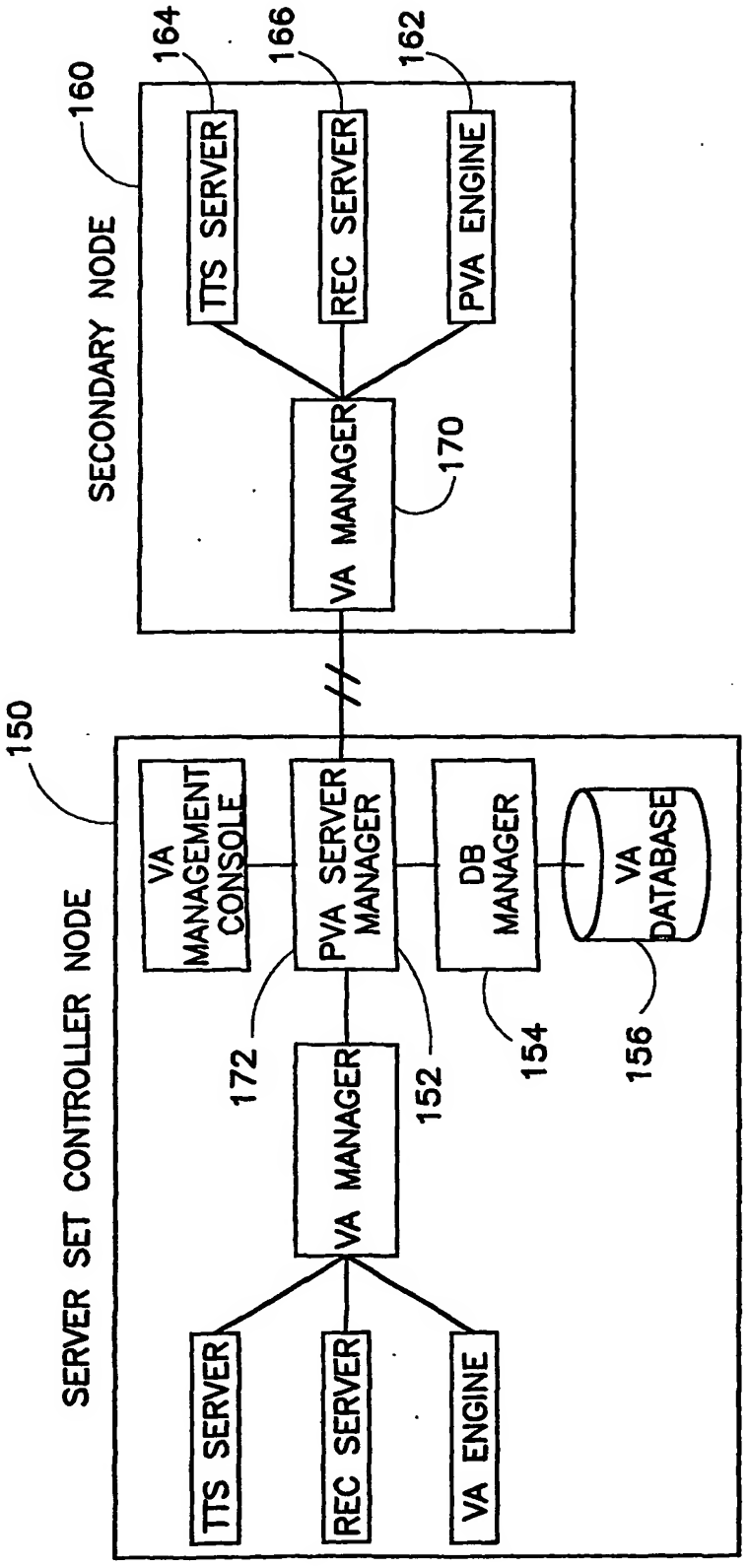

SECONDARY NODE
160
164
166
162
TTS SERVER
REC SERVER
PVA ENGINE
VA MANAGER
170
150
SERVER SET CONTROLLER NODE
VA MANAGEMENT CONSOLE
PVA SERVER MANAGER
DB MANAGER
VA DATABASE
172
152
154
156
VA MANAGER
TTS SERVER
REC SERVER
VA ENGINE

FIG. 8

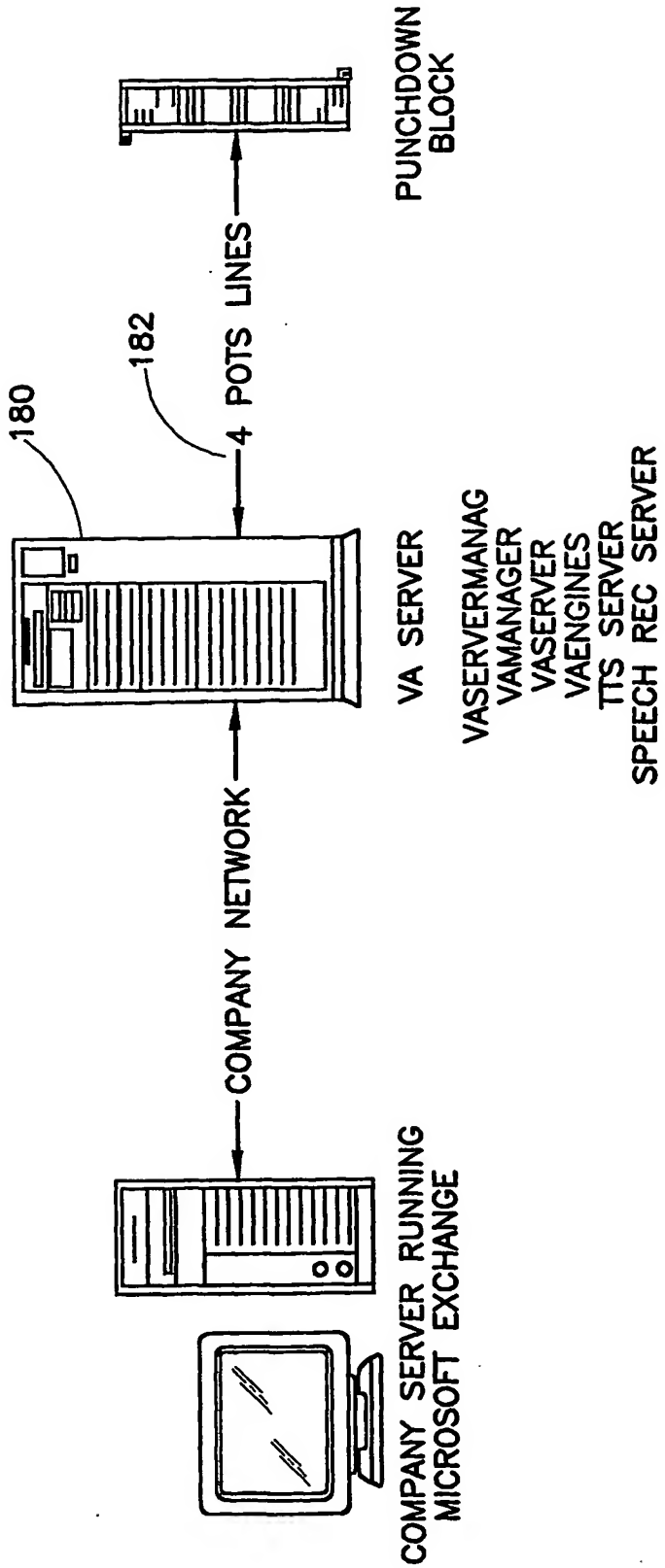

PUNCHDOWN BLOCK
4 POTS LINES
182
180
VA SERVER
VASERVERMANAG
VAMANAGER
VASERVER
VAENGINES
TTS SERVER
SPEECH REC SERVER
COMPANY NETWORK
COMPANY SERVER RUNNING MICROSOFT EXCHANGE

FIG. 9

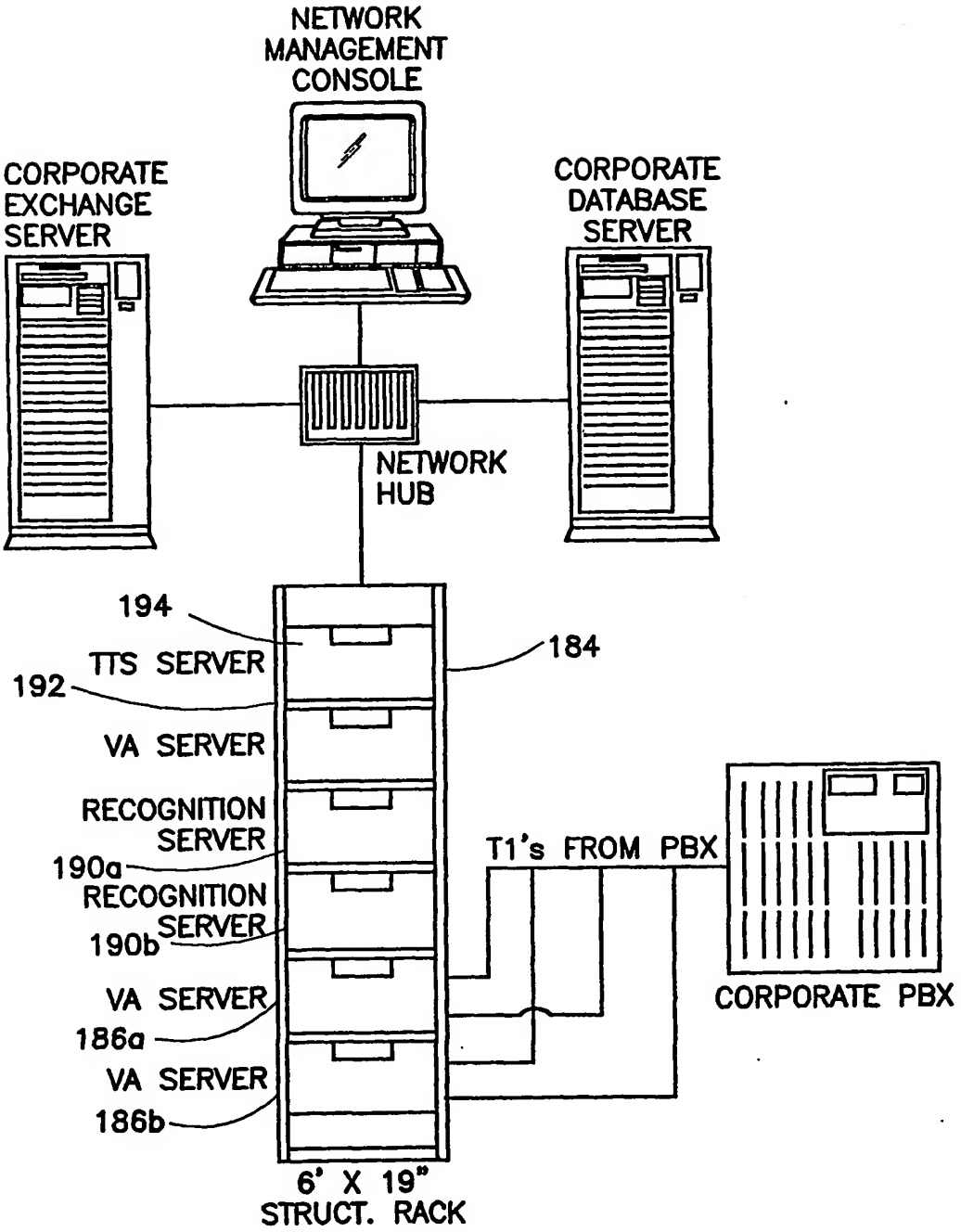

NETWORK MANAGEMENT CONSOLE
CORPORATE EXCHANGE SERVER
CORPORATE DATABASE SERVER
NETWORK HUB
194
TTS SERVER
192
VA SERVER
RECOGNITION SERVER
190a
RECOGNITION SERVER
190b
VA SERVER
186a
VA SERVER
186b
184
T1's FROM PBX
CORPORATE PBX
6' X 19" STRUCT. RACK

FIG. 10

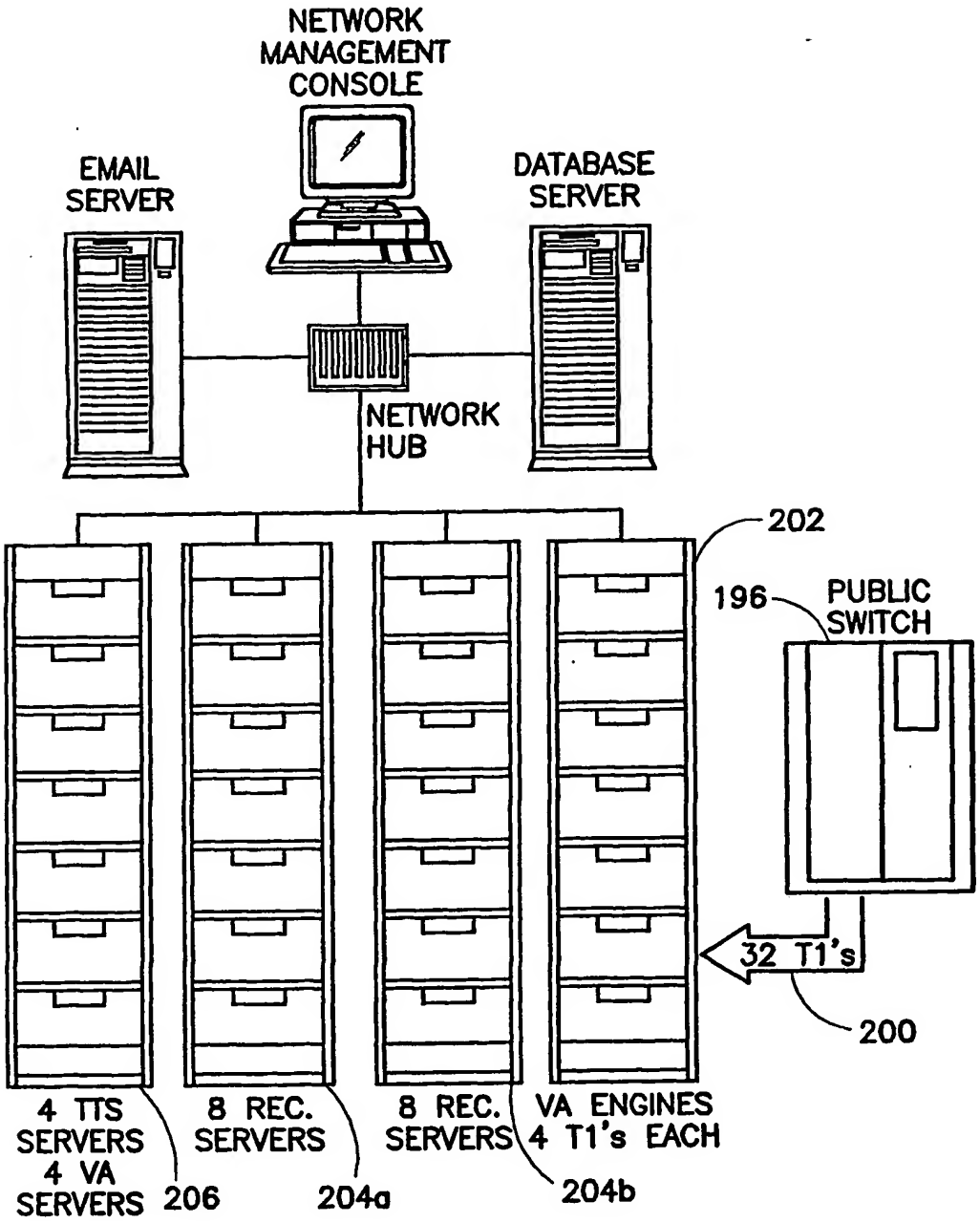

NETWORK MANAGEMENT CONSOLE
EMAIL SERVER
DATABASE SERVER
NETWORK HUB
202
196
PUBLIC SWITCH
32 T1's
200
4 TTS SERVERS 4 VA SERVERS
206
8 REC. SERVERS
204a
8 REC. SERVERS
VA ENGINES 4 T1's EACH
204b

FIG. 11

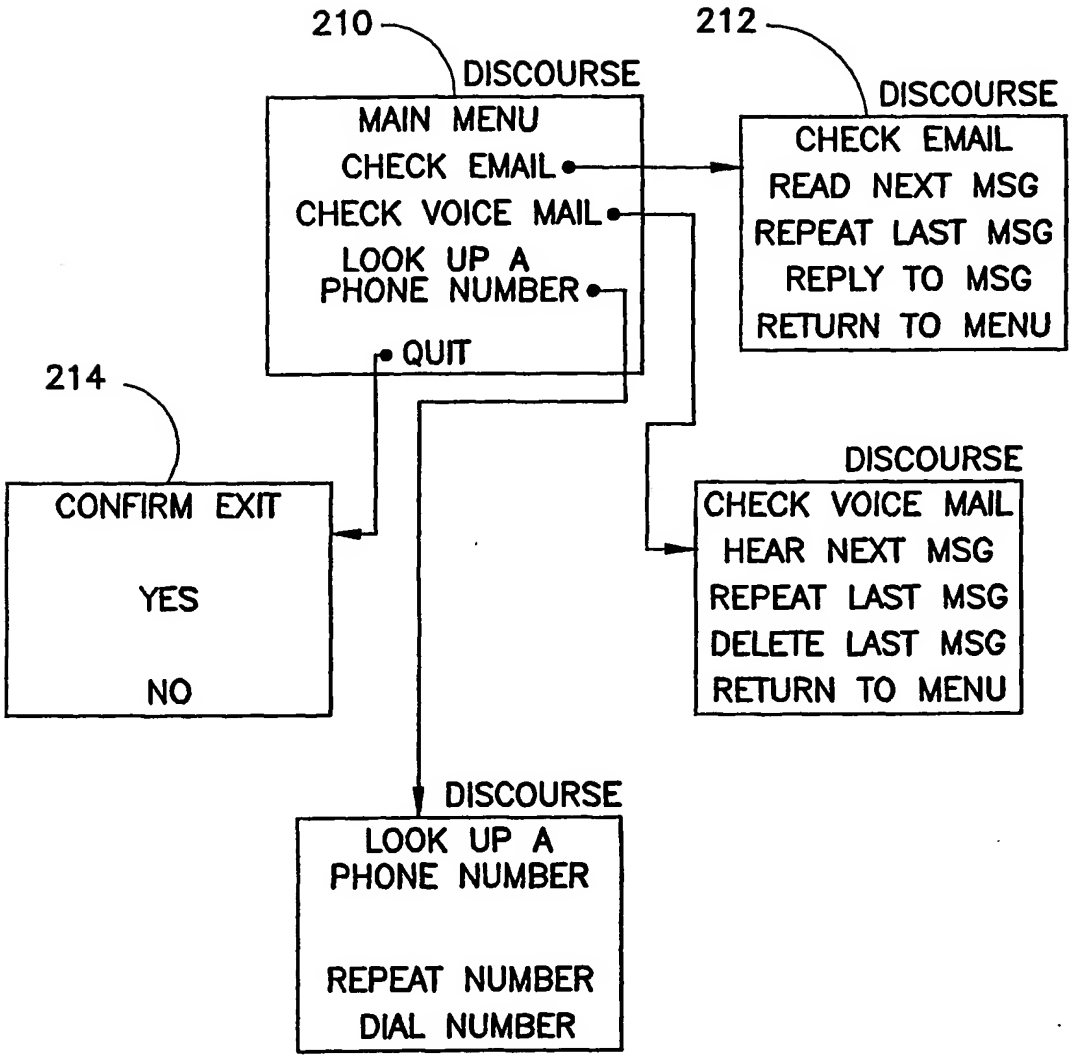

210
DISCOURSE
MAIN MENU
CHECK EMAIL
CHECK VOICE MAIL
LOOK UP A PHONE NUMBER
QUIT
212
DISCOURSE
CHECK EMAIL
READ NEXT MSG
REPEAT LAST MSG
REPLY TO MSG
RETURN TO MENU
214
CONFIRM EXIT
YES
NO
DISCOURSE
CHECK VOICE MAIL
HEAR NEXT MSG
REPEAT LAST MSG
DELETE LAST MSG
RETURN TO MENU
DISCOURSE
LOOK UP A PHONE NUMBER
REPEAT NUMBER
DIAL NUMBER

FIG.12

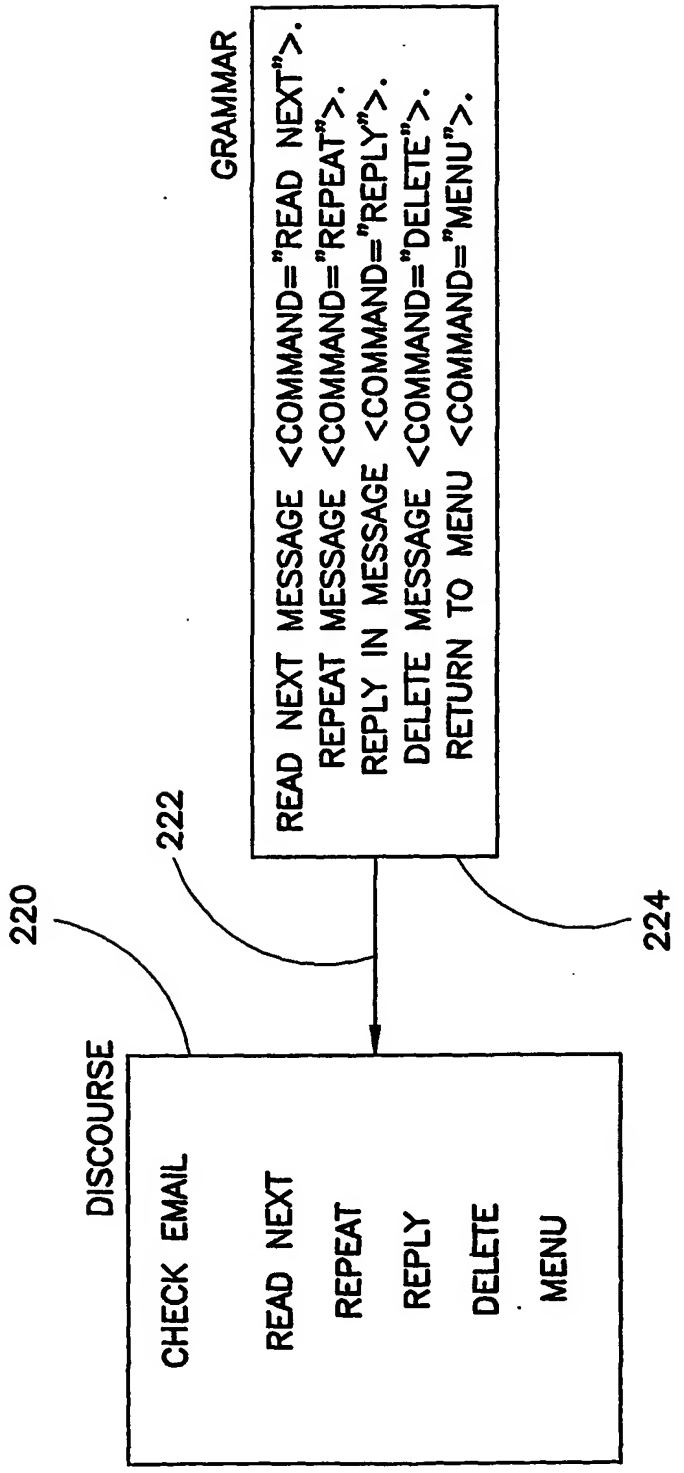

GRAMMAR
READ NEXT MESSAGE <COMMAND="READ NEXT">.
REPEAT MESSAGE <COMMAND="REPEAT">.
REPLY IN MESSAGE <COMMAND="REPLY">.
DELETE MESSAGE <COMMAND="DELETE">.
RETURN TO MENU <COMMAND="MENU">.
222
220
224
DISCOURSE
CHECK EMAIL
READ NEXT
REPEAT
REPLY
DELETE
MENU

FIG.13

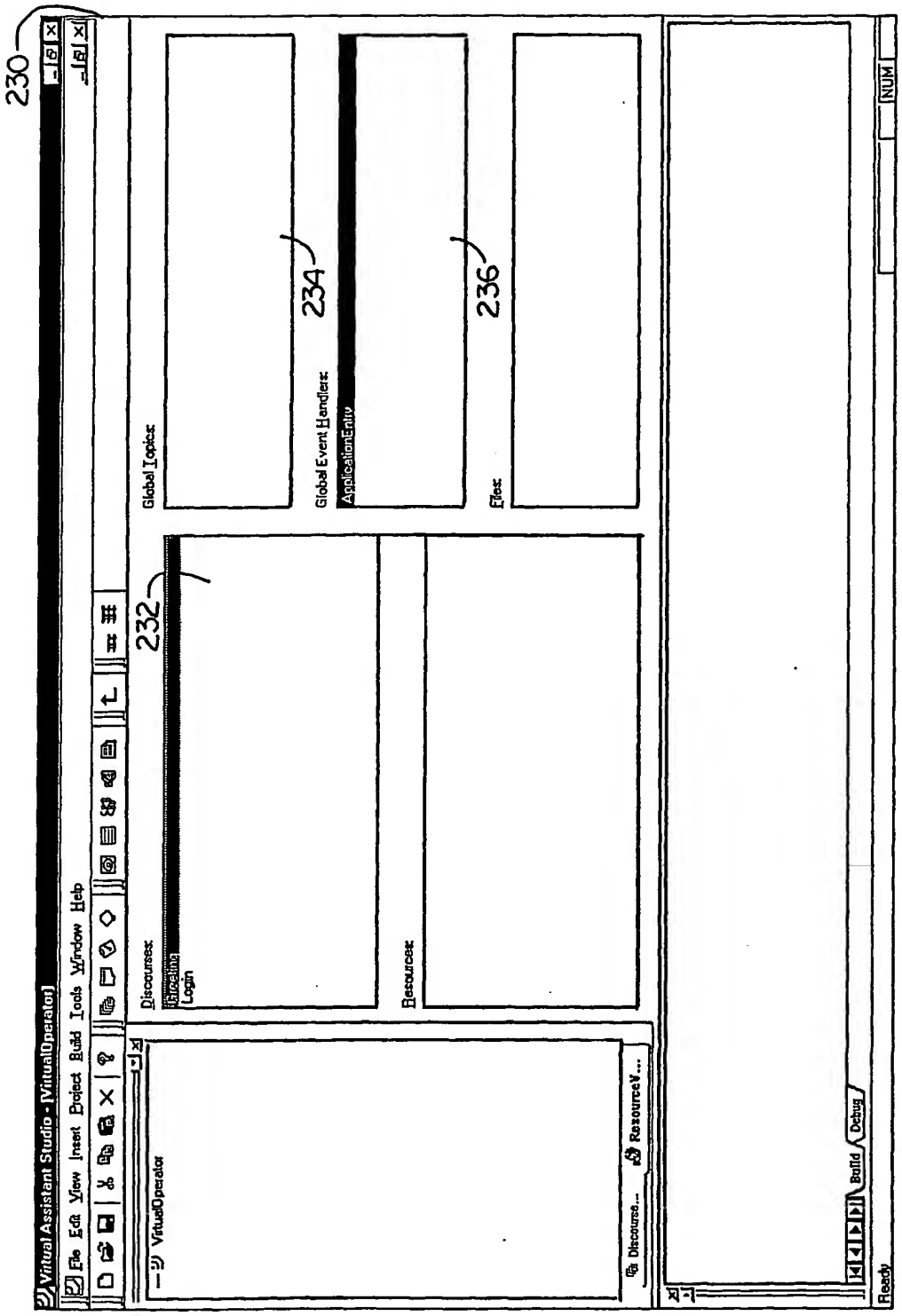

230
Virtual Assistant Studio - [VirtualOperator]
File Edit View Insert Project Build Tools Window Help
VirtualOperator
Discourses:
Allocating
Login
232
Resources:
Global Topics:
234
Global Event Handlers:
ApplicationEntry
236
Files:
Discourse...
ResourceV...
Ready
Build
Debug
NUM

FIG. 14

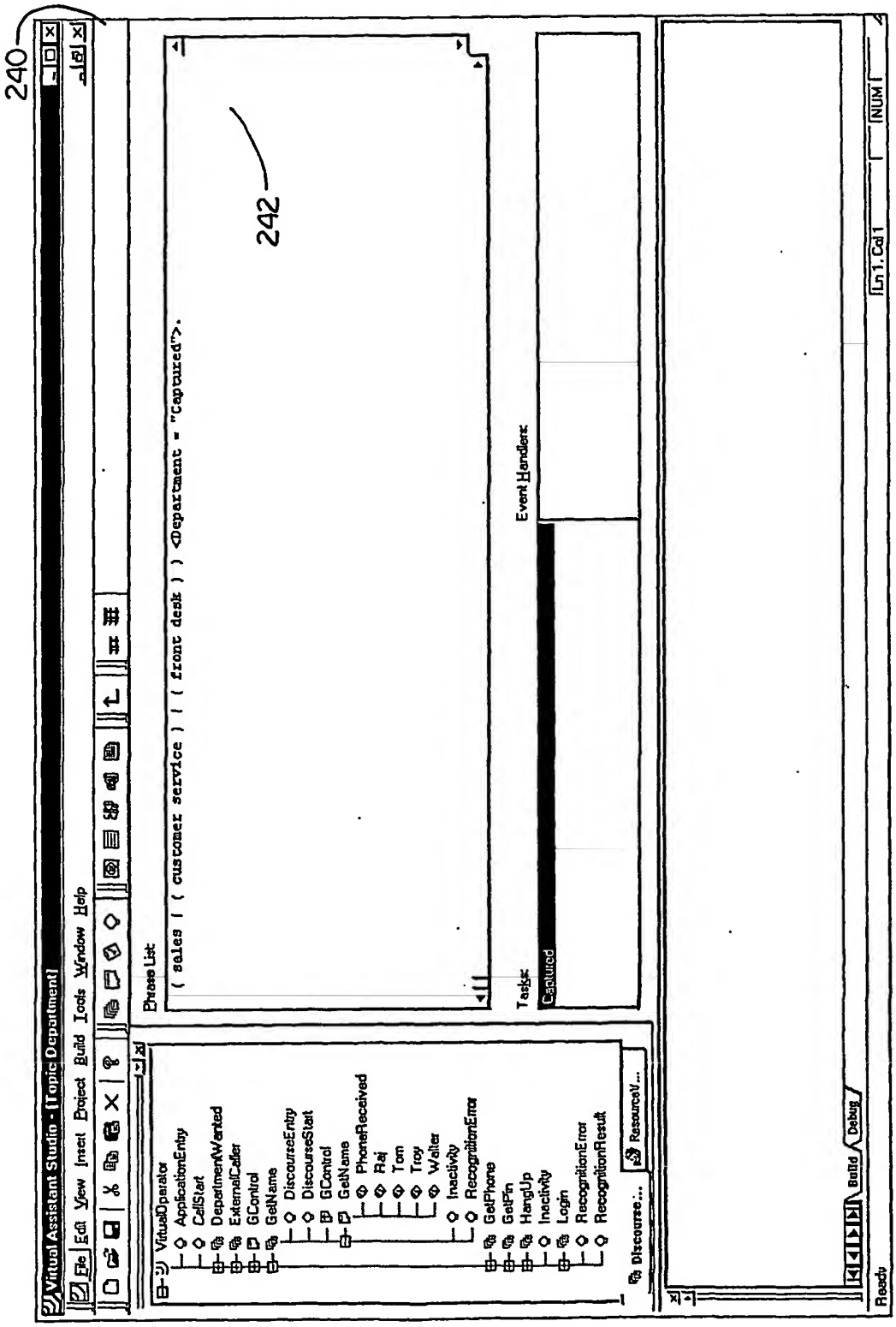

240
242
Virtual Assistant Studio - [Topic Department]
File Edit View Insert Project Build Tools Window Help
VirtualOperator
ApplicationEntry
CallStart
DepartmentWanted
ExternalCaller
GControl
GetName
DiscourseEntry
DiscourseStart
GControl
GetName
PhoneReceived
Raj
Tom
Troy
Walter
Inactivity
RecognitionError
GetPhone
GetPin
HangUp
Inactivity
Login
RecognitionError
RecognitionResult
Discourse...
ResourceV...
Phrase List:
(sales | (customer service) | (front desk)) <Department = "captured">.
Tasks:
Captured
Event Handlers:
Build
Debug
Ready
Ln 1, Col 1
NUM

FIG. 15

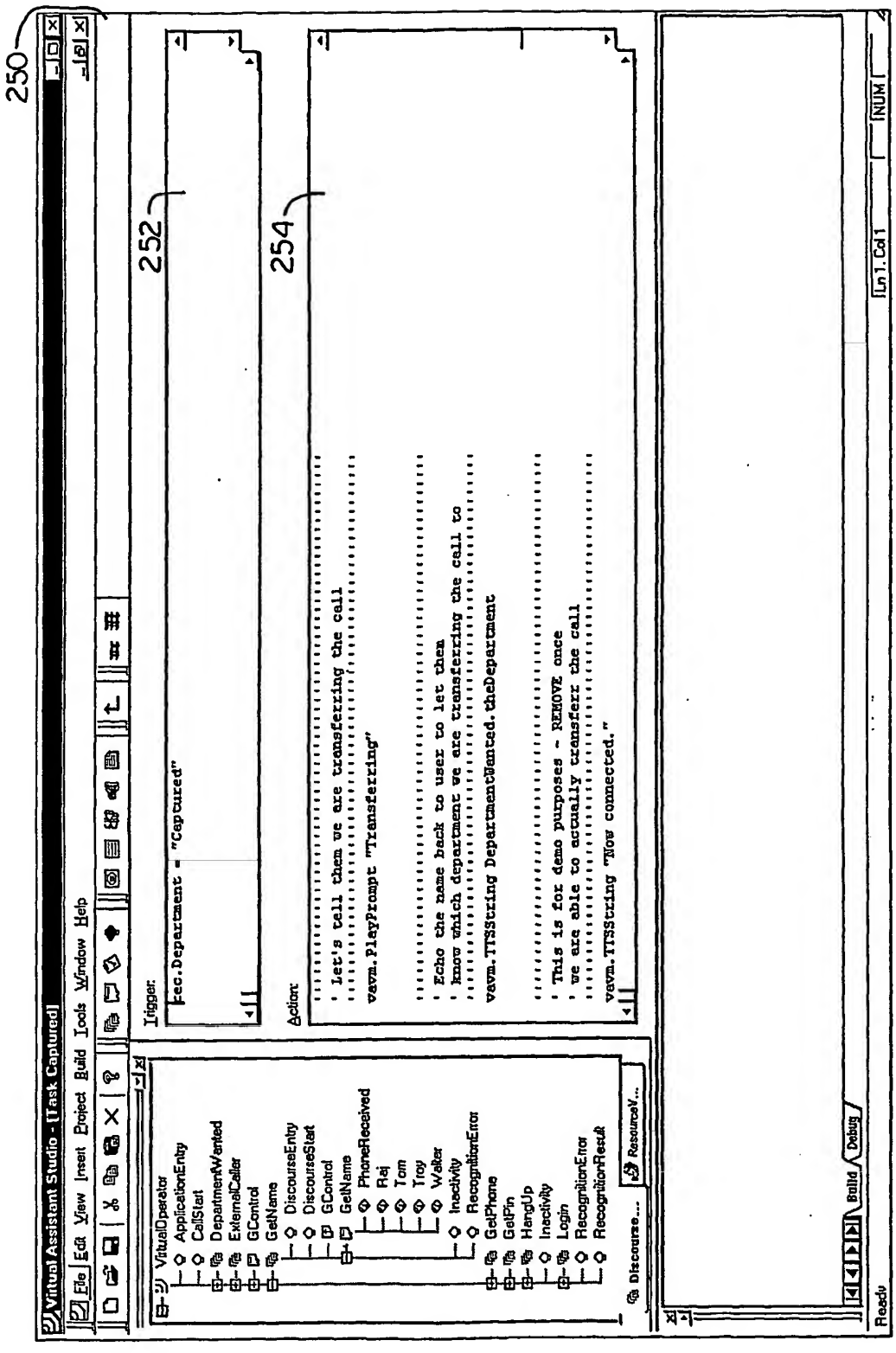

250
Virtual Assistant Studio - [Task Captured]
File Edit View Insert Project Build Tools Window Help
VirtualOperator
ApplicationEntry
CallStart
DepartmentWanted
ExternalCaller
GControl
GetName
DiscourseEntry
DiscourseStart
GControl
GetName
PhoneReceived
Raj
Tom
Troy
Walker
Inactivity
RecognitionError
GetPhone
GetPin
HangUp
Inactivity
Login
RecognitionError
RecognitionResult
Discourse...
ResourceV...
Trigger:
252
Rec.Department = "Captured"
Action:
254
' Let's tell them we are transferring the call
vavm.PlayPrompt "Transferring"
' Echo the name back to user to let them
' know which department we are transferring the call to
vavm.TTSString DepartmentWanted.theDepartment
' This is for demo purposes - REMOVE once
' we are able to actually transferr the call
vavm.TTSString "Now connected."
Build
Debug
Ready
Ln 1, Col 1
NUM

FIG. 16

FIG. 17

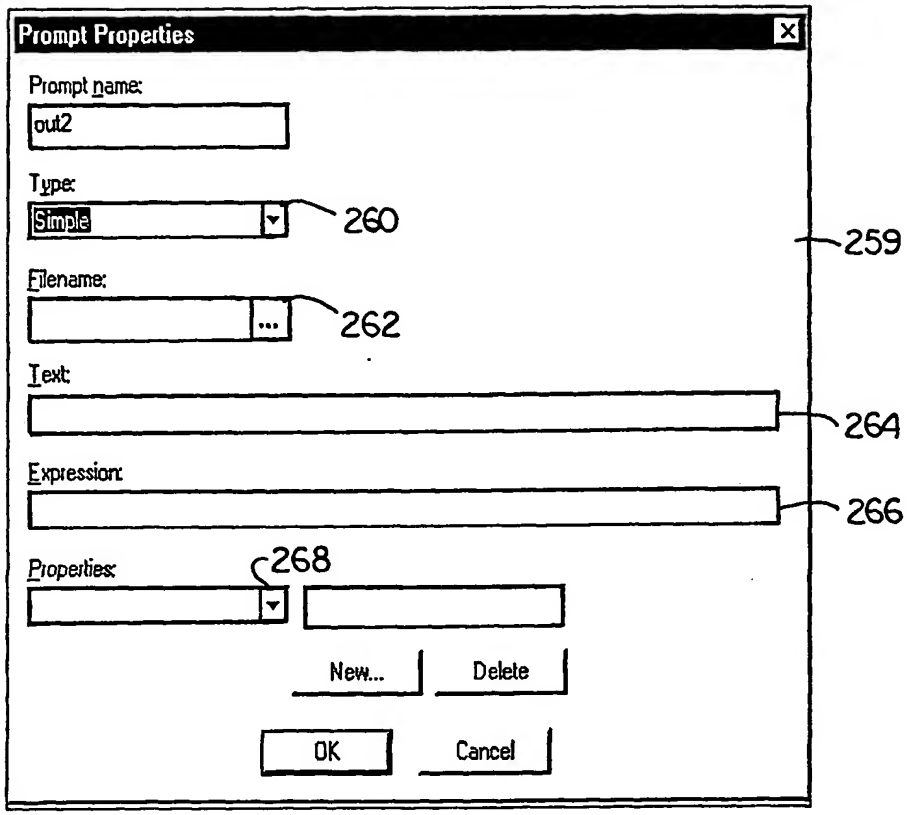

Prompt Properties
Prompt name:
out2
Type:
Simple
260
259
Filename:
262
Text:
264
Expression:
266
Properties:
268
New...
Delete
OK
Cancel

FIG. 18

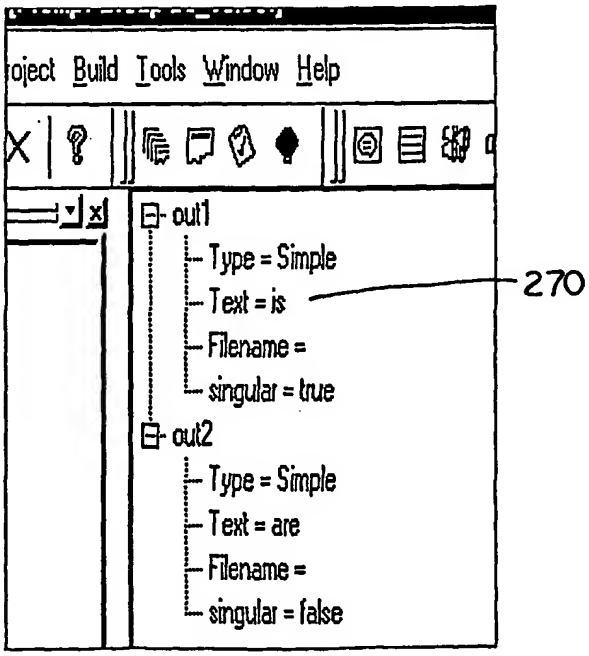
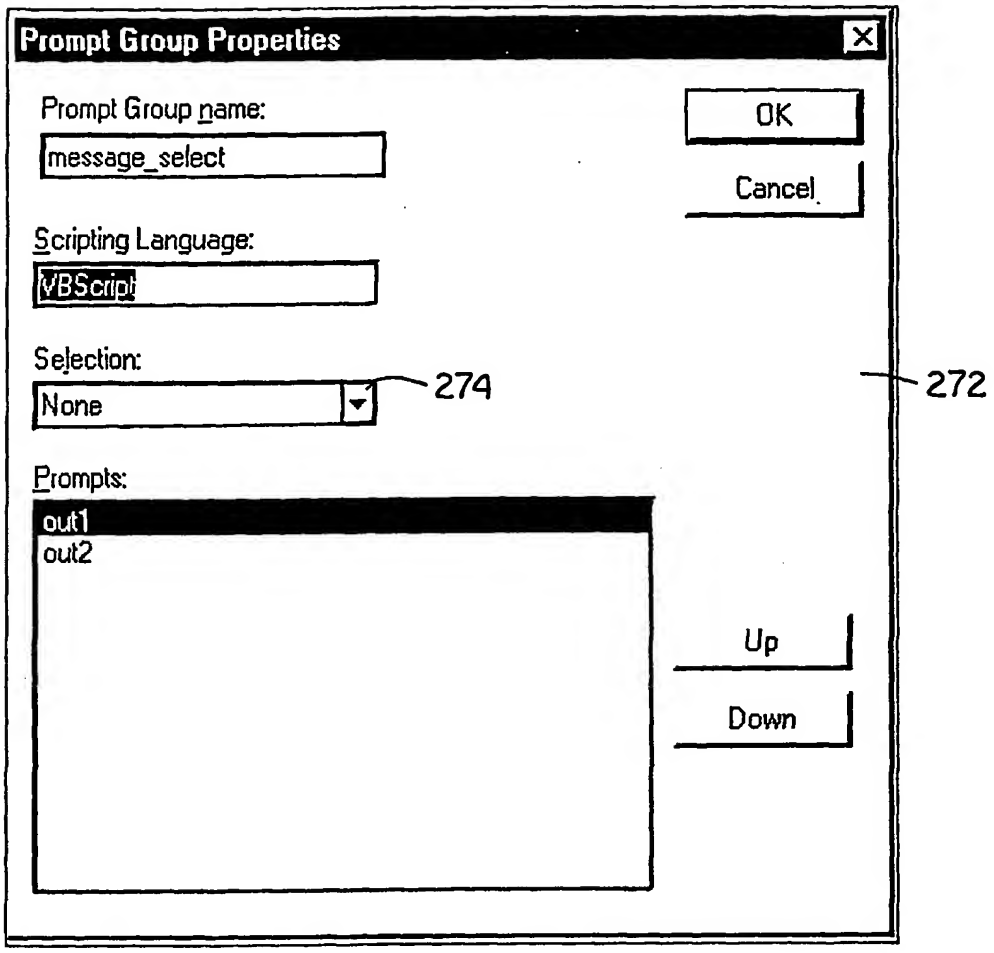

oject Build Tools Window Help
out1
Type = Simple
Text = is
Filename =
singular = true
out2
Type = Simple
Text = are
Filename =
singular = false
270

FIG. 19


Prompt Group Properties
Prompt Group name:
message_select
OK
Cancel
Scripting Language:
VBScript
Selection:
None
274
272
Prompts:
out1
out2
Up
Down

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/06882

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 9/45; H02H 3/05
US CL : 717/4; 714/38
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 717/4; 714/38

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,028,999 A (PAZEL) 22 February 2000 (22.02.2000), col.4-col.10.	1-41
Y	US 5,809,303 A (SENATOR) 15 September 1998 (15.09.1998), col.3-col.7.	1-41
Y	US 5,745,675 A (HERBIG et al.) 28 April 1998 (28.04.1998), col.15-col.16.	1-41

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 May 2001 (10.05.2001)	06 JUN 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. 703 305-3230	Authorized officer ALVIN OBERLEY Telephone No. 703 305-3665

Form PCT/ISA/210 (second sheet) (July 1998)